Universitat de Lleida

TREBALL FINAL DE GRAU

Estudiant: Víctor Vilaseca Villas

Titulació: Grau en Enginyeria Informàtica

Títol de Treball Final de Grau: **Aplicació per al seguiment de correus electrònics transaccionals**

Director/a: **Francesc Giné De Sola, Jordi Freixa Serra**

Presentació

Mes: Juliol

Any: 2020

Índex

Abstract

Ilerna Online is an internacional business exclusively dedicated to promote and support professional formation on the bases made up by and quality, using new techonologies and focusing on online learning. Ilerna Online is compound by several departments which lead it's path to the top.

The project titled "Aplicació per al seguiment de correus electrónics transaccionals" is impulsed internally by the IT department. Seeking the objective of real-time transaction *email* tracking, the IT department want an absolute control over all transactional mailing sent from the company.

Thus the main objectives of the project are to create a software that allows a new metric, with all the information about the *emails* that have been sent, to be created, studied and stored. In addition we want to create a tool that will be able to centralize all different sending points into one, and furthermore it will provide a visual way retrive all the data stored need to be implemented.

Resum

Ilerna Online és una empresa d'àmbit internacional, lider en Formació Professional *online*, dedicada a promoure una formació professionalitzadora i integradora basada en la innovació i el desenvolupament constant en els diferents àmbits utilitzant les noves tecnologies en tots els seus departaments. Això ha fet que sigui una organització puntera en l'àmbit de la formació a distància essent actualment un centre de referència.

El projecte a desenvolupar s'anomena "Aplicació per al seguiment de correus electrònics transaccionals", és un projecte consistent en la integració i utilització de diverses eines de programació per tal d'aconseguir un seguiment, a temps real, dels correus transaccionals enviats des de l'empresa, impulsat internament des del departament d'informàtica amb el que he col·laborat.

L'objectiu és crear un *software* que ens permeti disposar d'unes mètriques que continguin totes les dades necessàries per així poder analitzar el comportament en els receptors dels correus electrònics transaccionals. Aquesta eina permetrà centralitzar tots els diferents punts d'enviament de correu i gestionar-los. A més, ajudarà a la visualització i tractament de les dades aconseguides.

Agraïments

Voldria agrair la col·laboració del meu tutor de treball final de grau Francesc Giné De Sola per la seva exigència, el constant suport, l'orientació i la confiança d'acompanyar-me durant la trajectòria del projecte.

Tanmateix, també voldria anomenar a Jordi Freixa Serra, tutor de treball de final de grau des de l'empresa, per la seva paciència, actitud positiva i dedicació durant l'estada en l'empresa. Així com a tots els professionals del departament d'IT d'Ilerna Online, els quals amb l'ambient de treball, organització i habilitat, han aportat el seu gra de sorra en la realització del projecte.

I com no, a la família: els meus pares Lluís i Cristina amb el seu estímul, suport i estima inqüestionable; i al meu germà Robert per la seva visió alternativa. Així com totes les altres persones que formen part de la meva vida i d'una manera o altra, han aportat a la realització d'aquest projecte.

1 Introducció

Durant el període de pràctiques a Ilerna Online vaig estar implicat en diversos projectes interns molt enriquidors i que em van aportar molta experiència.

Vaig aprendre amb ajuda de l'equip d'IT d'Ilerna Online a resoldre eficientment les diverses tasques què sorgien en el departament així com programar amb llenguatges (PHP) que fins al moment no havia utilitzat.

A més a més, en l'empresa s'usen nous *frameworks* [3] (Laravel) [5] i es treballa mitjançant una metodologia anomenada *Agile*[1] que serveix per organitzar la feina, els recursos i el temps dinàmicament. Fet que em va resultar un repte molt interessant d'aprendre.

Llavors, quan em van suggerir fer el treball de final de grau amb ells després d'acabar les pràctiques, em vaig entusiasmar i van informar-me de què existia un problema real on creien que jo podria aplicar tots aquests coneixements que havia anat adquirint. La proposta consisteix en poder fer una aplicació web, que permetés trobar una informació que per ara era inexistent, tractant temes de *Big Data*, *data analítics*, bases de dades, entre moltes altres eines.

1.1 Ilerna Online: Àmbit d'Aplicació

En aquest apartat parlarem sobre l'empresa Ilerna Online per tal d'informar i familiaritzar al lector sobre aquesta, tal com hem mencionat anteriorment.

L'empresa Ilerna Online es dedica en exclusiva a la Formació Professional a distància, centrant els seus esforços en una formació de qualitat i personalitzada. Ilerna ofereix més de 18 cicles formatius i té més de 10.000 alumnes, amb diferents seus nacionals i internacionals i essent molt innovadora en tots els seus àmbits. Sobretot però, s'ha de destacar el seu caire tecnològic, un pilar fonamental de l'empresa que amb dedicació i esforç és capaç de portar a la realitat tots els grans reptes que els hi són plantejats, així com d'estructurar tota l'arquitectura tecnològica sobre la qual es sosté l'empresa.

Per tant, com es pot induir, la gestió dels recursos electrònics i informàtics resulta de vital importància. Per això Ilerna compta amb un departament d'IT, compost per 16 professionals encarregats d'aquesta responsabilitat, què treballant conjuntament amb altres departaments, aconsegueixen donar un servei excel·lent per a l'alumne.

1.2 Necessitat i motivació del projecte

En una empresa com Ilerna Online, proporcionar informació adequada a l'alumne és vital, per tal d'aconseguir-ho s'han d'utilitzar diversos mitjans de comunicació que poden ser molt variats. Un d'aquests mitjans és l'*email*, que a més de poder ser personalitzat és una eina de gran abast de difusió i de cost reduït.

L'*email* empresarial es pot classificar en tres tipus: transaccional, relacional i promocional. En el projecte, ens centrarem en la tipologia transaccional, aquest permet informar el consumidor de qualsevol acció realitzada pel mateix dins d'algun dels aplicatius de l'empresa en qüestió. Un exemple d'aquests casos seria l'*email* de confirmació de compra que arriba després de realitzar-la. Aquest *email* contindrà informació predefinida sobre el producte comprat, que s'enviarà a cada un dels usuaris que facin una compra d'aquest producte.

La motivació d'aquest projecte, sorgeix de la necessitat de gestionar un creixent nombre de correus electrònics transaccionals enviats des de múltiples punts de l'empresa d'on a més, es vol aprofitar la centralització feta per a l'explotació de la informació dels correus.

En primera instància els correus enviats des de l'empresa eren relativament pocs, ja que la plataforma web era també de menor mida. La gestió d'aquests correus llavors no suposava un major problema i el no tenir una automatització ni *software* propi que permetés analitzar cadascun dels correus enviats, no generava cap desavantatge.

Tot això però canvia a partir d'una ràpida expansió de l'empresa, cada vegada amb més estudiants i que a més va oferint més i més serveis. Això suposa que el nombre de correus es multipliquin, així com el nombre de punts des dels quals s'envien.

Aleshores, com aquest mètode de comunicació amb l'estudiant esdevé molt utilitzat, comença a ser rellevant la seva gestió i aprofitar per recollir totes les dades que siguin d'utilitat en aquest canal de comunicació. D'aquesta manera, s'ajudarà a millorar la relació amb l'estudiant, així com la possibilitat d'obtenir estadístiques per valorar l'efectivitat del mètode emprat.

1.3 Vinculació amb el grau

L'interès d'aquest projecte rau en el fet que permet treballar amb diverses tecnologies actuals com per exemple Amazon Web Services, Laravel i RabbitMQ enfocades a diferents propòsits, amb les quals aconseguirem crear una nova eina. Donat que la menció de grau que estic cursant és la menció en Enginyeria del *Software*, treballar amb diferents *softwares* per crear un aplicatiu nou és una idea que em va atreure des d'un principi. En la meva opinió, fer un projecte propi en un entorn real, és una gran pràctica per aprendre i adquirir el màxim de coneixements.

1.4 Objectius

En aquest apartat descriurem els objectius específics d'aquest projecte:

- Investigar i familiaritzar-me amb noves eines per tal de poder efectuar aquest projecte.
- Definir els requeriments del projecte.
- Dissenyar l'estratègia més òptima.
- Planificar i calcular el cost del projecte.
- Fer tests per a la implementació del codi i desplegar l'aplicació en el servidor propi de l'empresa.
- Integrar l'enviament actual d'*emails* transaccionals amb un sistema de cues.
- Integrar el sistema de cues amb el núvol, i definir mètriques que capturin la informació que vulguem recollir de la interacció de l'alumne amb l'*email*.
- Recollir informació de cada correu electrònic transaccional i guardar-la en una base de dades.
- Gestionar aquesta informació i mostrar-la per obtenir gràfiques i conclusions.

1.5 Estructuració del projecte

Aquesta memòria l'hem estructurat en diferents capítols:

- Introducció: s'explicarà el context del projecte, l'empresa i els objectius marcats.
- Eines utilitzades: es definiran les eines utilitzades i el seu motiu d'ús.
- Disseny del projecte: es detallaran tots els passos així com també tota la planificació per dur a terme el projecte.
- Implementació del projecte: s'especificarà com s'escriurà el codi que compondrà el nostre projecte, explicant cada un dels dos components del nucli d'aquest.

- Avaluació del projecte: avaluarem el resultat final així com la realització del projecte, comparant-lo amb els objectius i els requeriments.
- Conclusions i futur del projecte: finalment, en aquest capítol es valorarà la feina feta i impressions del treball realitzat.

2 Eines utilitzades

En aquest capítol, a causa de la gran quantitat d'eines utilitzades en el projecte, descriurem la funcionalitat bàsica de cadascuna així com per a què es fan servir.

2.1 Amazon Web Services

Amazon Web Services (AWS) [6] és un conjunt de serveis informàtics al núvol de l'empresa Amazon que permeten treballar amb multitud d'àmbits. Oferint des de tecnologies d'infraestructura, com de còmput, emmagatzemament, bases de dades, anàlisis i IoT. AWS ofereix multitud d'eines interconnectades entre si, que permeten crear tota classe d'aplicacions portant-les al núvol, fent que sigui més ràpid, fàcil i rentable.

Amazon Simple Notification Service (Amazon SNS)

Amazon Simple Notification Service (Amazon SNS) és un servei de missatgeria de publicació/subscripció completament administrada, d'alta disponibilitat, segur i amb una durabilitat que permet desacoblar microserveis, sistemes distribuïts i aplicacions sense servidor.

Amazon SNS proporciona temes per a la missatgeria d'alt rendiment, basada en la inserció i per a molts destinataris. Mitjançant l'ús de temes d'Amazon SNS, els sistemes de publicadors poden distribuir els missatges en un gran nombre de punts d'enllaços de subscriptors per al processament paral·lel, incloses diferents tipus de servei d'AWS, així com *webhooks* HTTP/S. A més, SNS es pot utilitzar per distribuir notificacions a usuaris finals mitjançant la inserció de mòbils, SMS i correus electrònics.

Amazon Simple Email Service (Amazon SES)

Amazon Simple Email Service és un servei d'enviament d'*emails*, basat en el núvol, dissenyat per ajudar als responsables de màrqueting digital i als desenvolupadors d'aplicacions a enviar correus electrònics de màrqueting, notificacions i transaccions. És un servei fiable i rendible per les empreses de totes les mides que utilitzin l'*email* per mantenir-se en contacte amb els clients.

2.2 Qlick

Qlick [7] és una aplicació de *bussiness intel·ligence* que busca entre diversos tipus de dades i les interrelaciona, permetent a l'usuari mostrar aquestes relacions que a vegades poden ser complexes, de manera clara i entenedora.

A més, permet fer tot això en temps real, navegant entre gràfiques per obtenir en cada moment la informació què es necessita. Qlick és capaç de treballar a més amb *Big Data*, compartir les solucions obtingudes i ajudar a la presa de decisions amb la potència de tota la informació recollida.

2.3 RabbitMQ

RabbitMQ [8] és un agent de missatges o *broker* de codi obert que implementa el protocol AMQP [2]. El servidor RabbitMQ està escrit amb el llenguatge de programació Erlang i construït amb l'entorn *Open Telecom Platform*. Les biblioteques RabbitMQ estan disponibles per a la majoria de llenguatges de programació. RabbitMQ va ser creat per Rabbit Technologies Ltd. El 2010 aquesta companyia va ser adquirida per SpringSource, una divisió de VMware.
El projecte RabbitMQ consta de diferents parts:

- El servidor d'intercanvi RabbitMQ.
- Passarel·les per als protocols HTTP, XMPP i STOMP.

- Biblioteques de clients per a Java i el *framework* .NET. (Biblioteques similars per a d'altres llenguatges també es troben disponibles).
- El *plugin Shovel* (pala) que s'encarrega de copiar (replicar) missatges des d'un corredor de missatges a d'altres.

2.4 Laravel

Laravel[5] és un *framework* de PHP que permet l'ús d'una sintaxi més refinada i expressiva per a poder crear codi de forma senzilla, ajudant a evitar el codi llarg i feixuc i permetent multitud de funcionalitats. A més, aprofita elements d'altres *frameworks* i utilitza característiques de les últimes versions de PHP. La seva estructura està formada per dependències, especialment Symfony [9], el que implica que el seu desenvolupament depèn molt d'aquestes dependències.

Característiques bàsiques de Laravel:

- Sistema de *ruteig restful*
- Motor de plantilles
- Peticions Fluent
- Eloquent ORM
- Basat en Composer
- Support per la caché
- Support per MVC
- Adopta les especificacións PSR-2 y PSR-4

2.5 Scrum

Per tal d'organitzar el projecte s'utilitzarà la metodologia Scrum, una metodologia específica que compleix amb el manifest *Agile* i és la més utilitzada arreu del món per a projectes de desenvolupament del *software*.

Scrum es diferencia sobretot de les altres metodologies *Agile*, per les iteracions curtes que fa anomenades *Sprint*, on s'entrega al client a cada iteració (*Sprint*) la feina que s'està realitzant. Les iteracions o *Sprints* en aquest projecte consistiran en dues setmanes laborals cadascun.

Scrum a més, també regula les reunions de l'equip o desenvolupadors amb esdeveniments propis com: *Daily meeting*, *Sprint Review*, *Sprint Retrospective* entre altres.

2.6 Justificació d'eines utilitzades

L'elecció d'eines en un projecte de *software* no és trivial, ja que hauràs de conviure amb aquestes durant, no només la realització del projecte sinó al llarg de tota la seva vida útil. Per tant, per tal de triar les millors eines per un projecte, es recomana tenir en compte almenys aquestes consideracions:

- Dificultat d'ús.
- Cost.
- Suport de la comunitat, facilitat per aprendre.
- Mestria en l'ús de l'eina per part del desenvolupador.
- Futur de l'eina utilitzada.

Una vegada considerats aquests punts, es pot fer l'elecció més encertada en el moment del temps que es pren la decisió. Però sempre s'ha de plantejar la possibilitat de migració parcial o total de les eines utilitzades.

En aquest cas, totes les eines elegides complien les consideracions plantejades anteriorment i per tant, podem concloure en que les eines utilitzades són adequades per implementar aquest projecte.

3 Disseny del projecte

El disseny del projecte és el procés d'elaboració de la proposta de treball d'acord a pautes i procediments sistemàtics, un bon disseny ha d'establir un diagnòstic de la situació; considerar diverses estratègies possibles per enfrontar-la, així com definir els objectius del projecte (generals i específics) i els recursos mínims necessaris.

3.1 Requeriments

El procés de reunió dels requeriments dins del procés d'anàlisi previ és indispensable per al correcte desenvolupament d'un projecte de *software*. Per tant, per obtenir els requeriments necessaris en aquest projecte, haurem de treballar conjuntament amb el Jordi Freixa, CEO d'Ilerna Online, d'on s'extraurà tot el que el projecte ha de complir, amb el propòsit d'arribar a un llistat estructurat i desgranat de les funcionalitats essencials del sistema.

3.1.1 Requeriments Funcionals

Els requeriments funcionals són els que s'encarreguen de definir el conjunt d'entrades, comportaments i sortides del programa; com també les funcionalitats específiques que se suposa que el sistema ha de complir:

- El sistema haurà de guardar tots els correus que s'enviïn per el mateix.
- El sistema haurà de tenir una interfície gràfica per poder visualitzar totes les dades.
- El sistema haurà de ser capaç de filtrar correus per tipus, correspondència i estat.
- El sistema haurà de permetre adjuntar fitxers i aplicar plantilles per als correus.
- El sistema haurà de permetre l'enviament a múltiples destinataris a més d'enviaments amb remitent ocult i àlies de remitent.

3.1.2 Requeriments No Funcionals

Els requeriments no funcionals són aquells que defineixen el que el *software* ha de complir en termes d'aparença, sensació, operabilitat i mantenibilitat:

- El sistema haurà de ser segur per tant, se seguiran totes les recomanacions d'ús del software i *frameworks* utilitzats.
- El sistema haurà d'estar preparat per estar sempre actiu.
- El sistema haurà de tenir una aparença coherent amb l'empresa.
- El sistema haurà de ser capaç de gestionar grans quantitats de dades.

3.2 Disseny

El disseny de programari és el procés de trobar solucions als requeriments esmentats. Una de les parts més importants del disseny és l'anàlisi de requisits de programari, que és una part essencial del procés de desenvolupament, ja que llista els requeriments que haurà de complir el programari.

Una vegada obtinguts els requeriments, fa falta organitzar el projecte per temporalitat i funcionalitats. Per tal de poder aconseguir una planificació i disseny adequats, en ser un projecte tant extens, és divideix en dues parts, *backend* i *frontend*.

La part del *backend* serà realitzada conjuntament amb un professional de l'empresa que em supervisarà per tal d'acabar implementant un *software* de qualitat. L'organització del *backend*, es

dividirà en cinc funcionalitats a completar en cadascun dels Sprints planejats.

La part del *frontend*, serà realitzada amb col·laboració amb els companys de l'empresa dedicats a aquestes tasques, ja que amb la seva experiència agilitzaran molt el procés.

Si el programari és completament automatitzat (no hi ha cap usuari ni interfície d'usuari), el disseny pot ser tan simple com un diagrama de flux o un text que descrigui una seqüència d'esdeveniments ja previstos.

Hi ha molts mètodes semi-estàndards (UML, per exemple) que ajuden a dur a terme el disseny de programari però, en qualsevol cas, quasi sempre es generarà una documentació del pla general per tal de tenir clares les bases del disseny.

En aquest apartat, s'explicarà utilitzant un diagrama de flux UML, diferenciat amb colors, cada una de les cinc parts principals del projecte i com interactuen. El color blau s'utilitza per visualitzar les eines o serveis que componen el projecte, el rosat, s'utilitza per diferenciar les parts d'integració que s'han hagut de fer des de zero per tal d'encaixar les peces.

- Projecte Core_Mail_Bus
- Mòdul RabbitMQ
- Projecte Core_Ilerna_Bus
- AmazonSES
- AmazonSNS
- Database

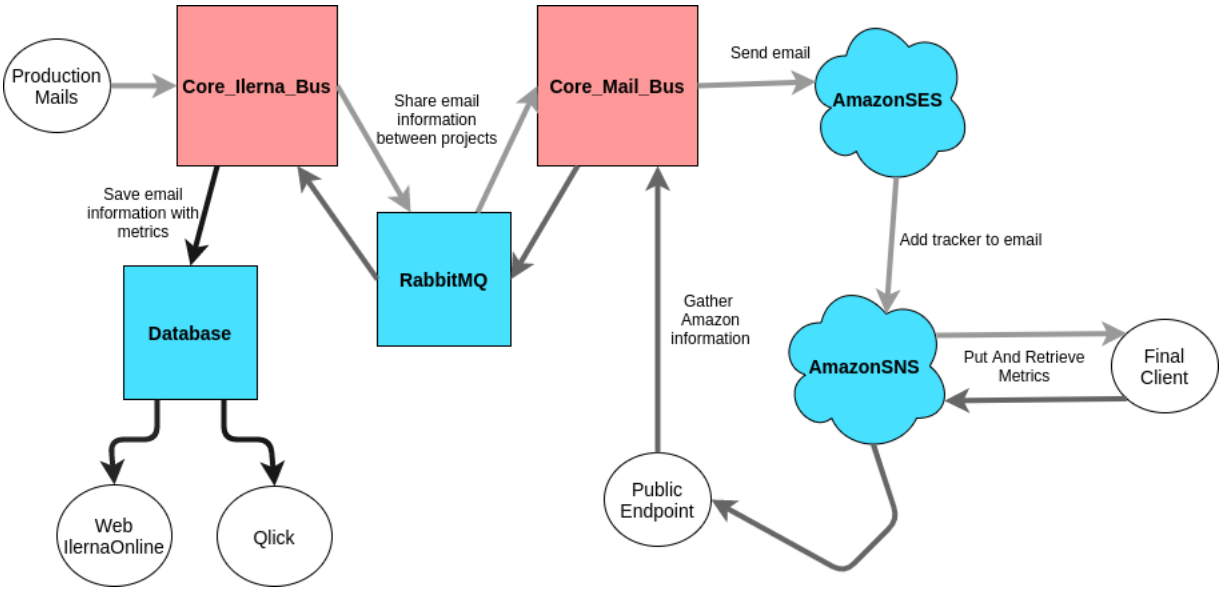


Figura 1: Diagrama de blocs del projecte

Core_Ilerna_Bus

S'encarrega de rebre la petició de correu que es vol enviar de "*Production Mails*", així com les dades de resposta dels serveis d'Amazon, que arriben a través de RabbitMQ des de Core_Mail_Bus, per tal de combinar la informació i guardar-la en la base de dades.

Core_Mail_Bus

Aquest bloc s'encarrega de comunicar-se amb tots els serveis d'Amazon i enviar la resposta a través de RabbitMQ a Core_Ilerna_Bus

AmazonSES

Aquest servei s'utilitza per l'enviament d'*emails*, aquest serà utilitzat durant el projecte mitjançant un SDK, el qual ens permetrà controlar aquest servei i disposar de tota la potència per l'enviament dels *emails* requerits.

AmazonSNS

Aquest servei genera notificacions d'Amazon que ens serveix per poder subscriure'ns a cada un dels correus que s'envien i rebre una notificació, aquesta, ens permetrà obtenir la informació més recent relacionada amb el correu en qüestió.

RabbitMQ

Situat entre Core_Ilerna_Bus i Core_Mail_Bus mencionats anteriorment, és un component clau del sistema que permetrà la comunicació entre els dos blocs, intercanviant tota la informació relativa als *emails*.

Database

Aquesta base de dades serà la que enregistrarà tots els correus i els seus esdeveniments associats. S'agafarà la informació per tal de poder-la visualitzar gràficament des de la web d'Ilerna Online i des de Qlick un aplicatiu de gràfiques a temps real capaç de treballar amb *Big Data*.

El tipus de base de dades que utilitzarem és MySQL, les característiques d'aquesta són: ser relacional, multi-fil, multiusuari i SQL (*Structured Query Language*). Hem triat aquest tipus de base de dades perquè no només és molt popular degut a la seva velocitat en executar consultes, sinó que a més suporta de forma nativa el llenguatge PHP.

Malauradament MySQL té certes limitacions, una d'elles, "no ser apta per grans quantitats de dades", aquesta restricció afectarà al projecte negativament en un estat avançat, però, en l'estat inicial en que ens trobem, no suposarà cap problema i fins que el volum de dades no augmenti exponencialment, no farà falta canviar la base de dades.

3.3 Temporalitat

Aquest projecte està pensat per tenir una durada de dos mesos d'implementació, un de revisió i un de documentació, durant els quals farem *Sprints* de dues setmanes. Amb aquests *Sprints* decidirem quina prioritat donar i quins passos seguir, per tal d'aconseguir el producte final.

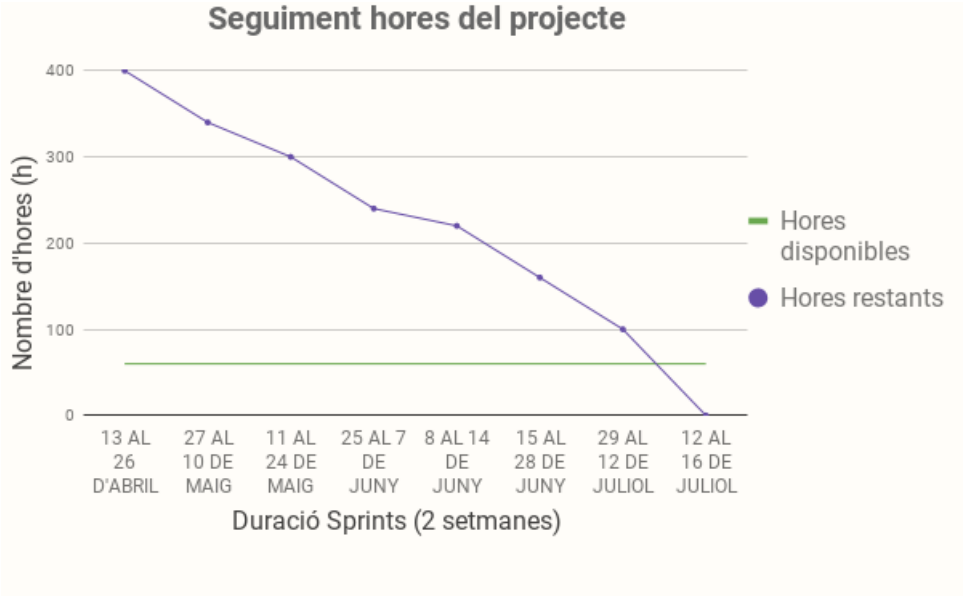


Figura 2: Progrés en hores del projecte

La figura 2 indica el progrés en hores al llarg dels *Sprints*. En aquesta es mostren les hores inicials del projecte en l'eix d'ordenades i la duració de cada un dels *Sprints* en l'eix abscisses. La línia blava de la figura indica el temps restant de les 400 hores que té el projecte al llarg dels *Sprints*. Per altra banda, la línia verda, indica el nombre màxim d'hores que es poden efectuar en cada *Sprint*, tenint en compte que per dia, es té planejat treballar 6 h.

Primer *Sprint*

El primer *Sprint* del projecte va ser d'investigació i preparació, es van buscar eines per a poder aconseguir les funcionalitats dels objectius marcats del projecte i una vegada trobades, es van fer "píldores informatives"[15] del seu ús.

Així mateix, es van fer els prototips de les funcionalitats que voldríem implementar amb un anàlisi de la dificultat d'ús, l'adaptació al projecte, el preu i suport de la comunitat.

A més, es van estudiar els requeriments inicials del projecte i es va fer una previsió de l'organització de treball.

Segon Sprint

En el segon *Sprint*, configurarem AMAZON SES aprofitant el seu entorn de prova i comprovarem que el destinatari rebi el correu sense problema. Per fer això, haurem d'habilitar el servei de proves, configurar la regió on ens trobem geogràficament, obtenir les claus SMTP i les credencials AWS, a més d'activar les adreces de les quals enviarem els correus amb àlies i grups.

Seguidament, després de comprovar el correcte funcionament AMAZON SES, configurarem AMAZON SNS, l'altre servei necessari d'Amazon. En aquest, haurem d'activar un *endpoint* públic així com crear les categories que vulguem per dividir el nostre correu, per poder afegir *tags* i temes. Finalment, habilitarem les diverses funcions creades que afectaran al correu i ens permetran rebre dades a temps real sobre el seu estat.

Tots aquests passos es faran inicialment en la pròpia interfície web d'Amazon sense haver de codificar res. D'aquesta manera podrem observar el funcionament i comportament de les eines utilitzades

i veure que la configuració utilitzada és la correcta.

Tercer Sprint

En finalitzar de configurar els serveis d'Amazon, continuarem amb el funcionament de RabbitMQ. el nostre gestor de cues, aquest programa s'utilitzarà per organitzar la comunicació d'*emails* en el nostre projecte, així com de *buffer* de la informació que els correus aniran intercanviant usant els nostres sistemes.

Per tal de fer funcionar aquest *software* s'instal·larà localment en la màquina, ja que així es testejarà bé el seu funcionament per separat passant-li qualsevol tipus d'informació no rellevant amb diverses freqüències i pes en Mb.

És molt important tenir en compte certes recomanacions i configuracions a l'hora d'utilitzar RabbitMQ tal com diu a la guia d'ús. [3]

Així doncs, algunes recomanacions que hem aplicat i realment afecten el rendiment són configurar els correus com a *push*, ja que si no acabaríem esgotant els recursos de la CPU, el que implicaria demanar constantment informació. Per bones pràctiques, s'ha d'intentar mantenir els canals i les connexions de RabbitMQ obertes el major temps possible, ja que així és més eficient i a més, evitar tenir moltes cues o cues molt llargues. D'aquesta manera evitarem utilitzar el disc dur i gestionar amb la RAM el màxim d'informació possible fet que optimitzarà el rendiment del sistema.

Quart Sprint

Una vegada obtingudes aquestes dues funcionalitats, el següent pas serà configurar dos nous projectes Laravel amb PHP [8]. El primer (Core_Mail_Bus) serà el controlador que utilitzarà AWS SDK[12] per comunicar-se correctament amb Amazon, servir els nostres correus i rebre la informació que Amazon ens torna. I l'altre, (Core_Ilerna_Bus) serà el que s'encarregarà de guardar aquesta informació, gestionar la base de dades i servir les dades a la web de l'empresa mitjançant una API.

Aquests dos projectes s'anomenaran Core_Ilerna_Bus i Core_Mail_Bus respectivament, i es comunicaran mitjançant el servei RabbitMQ.

La decisió de fer dos projectes Laravel separats per gestionar totes les funcionalitats del projecte, va ser avaluada al començament del tercer *Sprint*, per un motiu de creixement i evolució del *software*. Gràcies a aquesta separació obtenim avantatges palpables com:

- Desacoblar dues funcionalitats diferenciades.
- Permet la modificació de les parts mantenint la cohesió.
- El projecte pot utilitzar diferents tecnologies i eines.
- El projecte pot escalar eficientment i seguim el principi Obert-Tancat [10].

Cinquè Sprint

Per tal de tenir un *software* de qualitat, una vegada acabada la primera versió del projecte, aquesta s'ha de refinar, depurar i testejar. Això és el que es farà en aquest cinquè *Sprint*.

El primer pas que s'ha seguit en aquest projecte és fer proves de funcionalitat, aquestes, consisteixen en la monitorització exhaustiva de totes les funcionalitats sota qualsevol condició possible. Durant la realització d'aquestes, es van descobrir diversos conflictes amb marge de millora:

- Problemes amb la grandària dels adjunts.

- Problemes amb el nombre de caràcters del correu.
- Problemes de consum de memòria de RabbitMQ.
- Problemes de sincronització.
- Problemes de codificació de caràcters

Per resoldre les incidències es va haver de configurar de nou la base de dades per obtenir camps més grans, modificar la configuració especifica de RabbitMQ per disminuir el consum de memòria. En un anàlisi posterior, es va veure que els problemes de sincronització no eren culpa de la nostra implementació, si no dels serveis d'Amazon que tenen una latència elevada. I respecte la incidència en la codificació vam haver d'utilitzar UTF-8 compatible amb tots els caràcters llatino-americans.

Conseqüentment, es van resoldre tots els problemes, a excepció dels que depenien de tercers.

El següent pas, van ser proves d'integració per poder configurar i provar el comportament de l'aplicatiu en l'entorn de producció real. Aquestes, van ser favorables i com s'esperava el seu comportament va ser el predeterminat.

A més, per ajudar a la mantenibilitat del *software*, és van reescriure funcions amb l'objectiu de fer-les més curtes i entenedores. I també es va fer una documentació API utilitzant el referent d'Open API[11].

Sisè Sprint

L'últim pas en el desenvolupament del projecte és la creació d'un document que recull i especifica detalladament l'aplicació creada. Aquest document permetrà a qualsevol persona la comprensió i reproducció del *software* creat.

Crec que cal remarcar la importància d'aquesta documentació, ja que moltes vegades en la indústria del *software* no es reporta amb claredat el codi implementat. Aquest és un error greu que perjudica a tots els desenvolupadors per igual i ens dificulta el manteniment i posteriors cicles de treball.

Per tant, en aquest projecte, intentarem fer una documentació el màxim de curosa i detallada possible, centrant-nos sobretot en la comprensió i difusió dels continguts treballats.

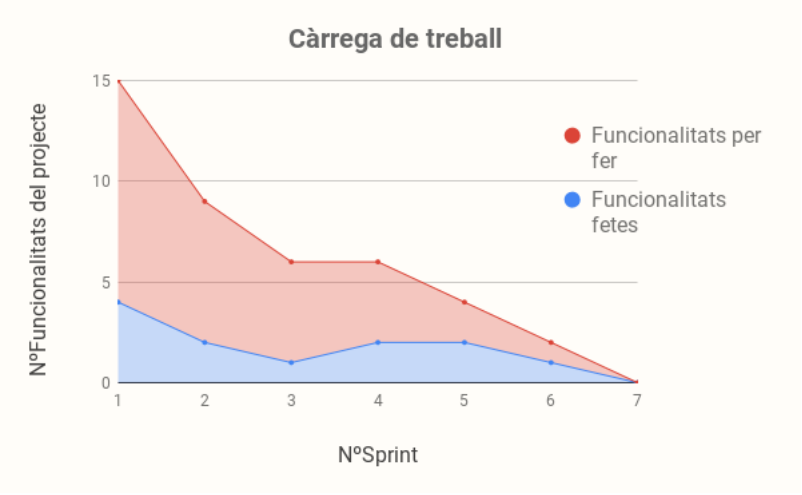


Figura 3: Feina acumulada

La figura 3, ens mostra un gràfic on l'eix d'ordenades indica el nombre de funcionalitats restants del projecte (roig), i l'eix d'abssisses el nombre de funcionalitats realitzades al llarg d'un *Sprint* (blau). Comparant aquestes dues variables, es pot observar l'avanç en funcionalitats del projecte.

3.4 Anàlisi de costos

L'anàlisi de costos és el procés d'avaluació dels recursos principals per realitzar un treball o un projecte. Aquest anàlisi estableix la qualitat i quantitat dels recursos i avalua els costos en termes econòmics per a les despeses que es generin en l'exercici del projecte, amb l'objectiu de determinar la seva viabilitat.

Per fer aquest estudi tan aproximat com puguem, haurem de fer unes consideracions prèvies. Primer, agafarem el sou mitjà d'un programador informàtic al llarg de l'any (25 euros/h) per aproximar el màxim possible a la realitat. A més contarem un dia laborable com 6 hores al dia, ja que no estava a temps complet en l'empresa.

DESCRIPCIÓ	Quantitat(h)	PREU PER UNITAT (€)	COST
Hores treballades:			
Planificació - 40 hores	40	25.00€	1,000.00€
Disseny - 60 hores	60	25.00€	1,500.00€
Prototipatge - 20 hores	20	25.00€	500.00€
Configuració d'eines - 40 hores	40	25.00€	1,000.00€
Implementació - 120 hores	120	25.00€	3,000.00€
Posada a producció - 20 hores	20	25.00€	500.00€
Testeig - 20 hores	20	25.00€	500.00€
Refinament - 40 hores	40	25.00€	1,000.00€
Documentació- 40 hores	40	25.00€	1,000.00€
Eines Utilitzades:			
Amazon Web Services - Per correu	1	0.00€	0.00€
Servidor 4GB - mes	1	40.00€	40.00€
Eines desenvolupament	1	35.00€	35.00€
	SUBTOTAL		10,075.00€
	Impost	21.00%	2,115.75€
	TOTAL		**12,190.75€**

Figura 4: Cost orientatiu del projecte

El preu final és de 12.190,75 euros. Aquest projecte, està indicat principalment per grans empreses, ja que aquestes pateixen sovint una necessitat de gestionar un nombre ingent de correus i els hi resulta atractiu poder gestionar així com obtenir més informació dels seus clients. En canvi possiblement aquest projecte no sigui l'indicat per PIMES, ja que segurament aquestes no tenen grans volums de correu i per tant el cost-benefici no els hi resultaria favorable.

4 Implementació del projecte

Els blocs Core_Ilerna_Bus i Core_Mail_Bus són les peces que composen el nucli del projecte, s'encarreguen del gruix de la integració de les eines així com de la seva comunicació. La implementació d'aquests dos blocs és el que serà tractat a continuació.

4.1 Core_Ilerna_Bus

Aquest bloc implementat amb PHP, utilitzant Laravel, és l'encarregat de la gestió i comunicació amb la base de dades i les eines de visualització. Aquest, està format per cinc mòduls principals:

- **Routes:** En aquest modul és on se situen el conjunt d'adreces navegables del bloc.
- **Models:** És on es definiran les taules de la base de dades així com la configuració d'aquesta.
- **Providers:** En aquest directori es defineixen els serveis executables, que ens permeten la comunicació amb les eines externes mencionades en el capítol 2.
- **Controllers:** És on es troba l'API que hem definit per tal d'obtenir la informació dels correus emmagatzemats.
- **Commands:** En aquest modul es troba la creació de comandes personalitzades que utilitzarem durant el projecte.

4.1.1 Routes

Routing or router, en desenvolupament web, és un mecanisme on les crides HTTP, són dirigides al codi que les tracta. Per simplificar-ho, l'arxiu que conté rutes, li determina el que hauria de passar quan un usuari visita una certa pàgina.

Totes les rutes del framework Laravel es defineixen als fitxers de la ruta, que es troben al directori de rutes. Aquests fitxers són carregats automàticament pel framework. Les rutes en el fitxer *routes\api.php* les hi assigna el grup de *middleware* API.

El *middleware* proporciona un mecanisme per filtrar les sol·licituds HTTP que entren a la nostra aplicació. Per exemple, Laravel inclou un *middleware* que verifica si l'usuari de la nostra aplicació està autentificat. Si l'usuari no n'està, el *middleware* redirigirà l'usuari a la pantalla d'inici de sessió. Tanmateix, si l'usuari està autenticat, el *middleware* permetrà que la sol·licitud accedeixi a l'aplicació.

Les rutes de Laravel més senzilles consisteixen en una URI(referencia) i una crida de retorn de tancament. Com per exemple:

```
<?php
Route::get('/', function()
{
return 'Hello World';
})
?>
```

En aquest bloc, tenim dos fitxers principals que detallarem a continuació.

4.1.1.1 channels.php

Aquesta classe com es necessita per autoritzar usuaris en canals privats i s'utilitza per a la comunicació de RabbitMQ que tenen els dos projectes.

Aquest fitxer només conté una ruta en el seu interior tal com es pot veure en 6, la ruta esmentada, s'encarrega de rebre una petició HTTP del client i autoritza a l'usuari a accedir a aquest canal. Retornant l'identificador de l'usuari que s'ha autoritzat a accedir.

4.1.1.2 api.php

Aquesta classe conté les diferents crides les quals componen l'API del projecte. Aquestes crides, permeten als companys de *frontend* obtenir la informació necessària per publicar en la pàgina web. Les crides desenvolupades són les següents:

- **Obtenir informació usuari:** S'encarrega de rebre una petició HTTP del client de tipus GET i retorna la informació de l'usuari.
- **Obtenir llista d'*emails* amb tots els seu detalls:** S'encarrega de rebre una petició HTTP del client de tipus GET i retorna una llista dels *emails* amb els seus detalls.
- **Enviar *email*:** S'encarrega d'enviar una petició HTTP de tipus put i envia un nou correu.
- **Afegir i descarregar adjunts:** S'encarrega de rebre una petició HTTP del client de tipus GET i descarrega l'arxiu adjunt d'un *email*, o permet afegir un adjunt a un correu amb una petició de tipus POST.

Per tal de poder veure la implementació d'aquestes consultar l'apartat 6 del annex.

4.1.2 Models

Totes les classes següents són extensió de la classe Model, això permet definir les estructures que la nostra base de dades seguirà, així com el tipus d'informació què es guardarà. Cada classe contindrà una variable protegida que especificarà el nom de la taula que usarem a la base de dades.

A més, el funcionament de Laravel permet declarar variables $hidden o $fillable, depenent de si les volem ensenyar o amagar segons ens interessi.

4.1.2.1 Apikey.php

Aquesta classe conté un camp booleà anomenat timestamps que, a l'estar en false, desactiva el seguiment del temps. I s'usarà per guardar les claus de l'API que fem servir.

4.1.2.2 Attachment.php

Aquesta classe serà utilitzada exclusivament per guardar informació. Que com el nom de la classe indica, es guardaran els adjunts que s'enviïn en els correus.

4.1.2.3 Email.php

Aquesta classe conté variables de tipus protected fillable que són els camps que ensenyarem i tindran els nostres *emails*, aquests són: 'id', 'ses_id', 'category', 'from', 'recipient', 'subject', 'body_plain', 'body_html', 'error_message', 'completed_at', 'from_name', 'cc', 'bcc'.

A més, la classe conté dues funcions, la primera d'elles s'encarrega de marcar el moment en què un correu es considera completat, i la segona permet aconseguir l'estat del correu al qual ens referim.

Aquesta segona funció relaciona els *emails* amb els EmailsEvents amb una relació OneToMany permetent saber a l'aplicació si un correu està Pendent, Obert, Enviat o té algun estat desconegut.

4.1.2.4 EmailEvent.php

Aquesta classe conté variables de tipus protected fillable i s'encarregarà de guardar la informació dels esdeveniments dels correus que s'enviïn.

Per tant, guardarà els camps que considerarem que un event d'*email* pot tenir, i seràn aquests: 'ses_id', 'event_type', 'link'.

4.1.2.5 IlernaMessage.php

Aquesta classe conté variables de tipus protected fillable i s'encarregarà de guardar la informació dels missatges que s'enviïn per RabbitMQ.

A més, la classe conté dues funcions, una que marca la data de quan s'ha rebut el missatge, i una altra que marca quan s'ha completat l'enviament del missatge.

4.1.2.6 User.php

Aquesta classe és extensió de la de la classe Autenticable i també té dues variables *hidden* per tal d'amagar el contingut que no volem i variables fillable per mostrar el contingut que desitgem.

4.1.3 Providers

Els proveïdors de serveis donen les funcionalitats de comunicació necessàries pel projecte. En aquesta secció definirem tots els proveïdors de serveis personalitzats que usarem configurats utilitzant el llenguatge de PHP i les llibreries de les eines esmentades anteriorment.

4.1.3.1 RabbitBusProvider.php

Aquesta classe conté les següents funcions esquelet i es responsabilitza d'oferir un servei per tal d'utilitzar RabbitMQ tal com es pot veure 6.

- **process_message():** processa un missatge cridant a la funció *exec* i indica per pantalla la rebuda d'un missatge.
- **createChannel():** crea un canal de RabbitMQ "donada una connexió vàlida" i permetrà la comunicació entre projectes.
- **destroyChannel():** tanca el canal de RabbitMQ el qual es passa com a paràmetre; d'aquesta manera s'alliberen recursos. Aquesta funció gestiona la creació d'instàncies controlant tots els possibles casos i actuant respectivament en cada cas.
- **getRabbitMqMessageResponse():** processa un missatge que estigui a la cua de RabbitMQ del canal response_queue i queda en espera fins que i hagin més missatges a processar.
- **injectRabbitMqMessage():** donat un missatge, es crea una canal de comunicació i llavors es publica aquest missatge en el canal. El missatge quedarà enregistrat i el canal serà tancat per tal d'alliberar recursos.
- **register():** serveix per relacionar Email i EmailEvent amb el servei que estem definint.

4.1.3.2 EventServiceProvider

La classe EventServiceProvider.php es dedica exclusivament a registrar *listeners* a certs esdeveniments perquè s'activin quan aquests ocorrin.

En aquest cas definim manualment la relació, i per funcionar s'hauria d'utilitzar la comanda *php artisan event:generate* al terminal així doncs, creem una relació entre la classe *Registered* del propi *Framework* i el nostre *listener*. La implementació es pot veure a 6

4.1.4 Controllers

Aquest apartat està dedicat a les classes de controladors, aquestes sempre seràn extensió de *Controller*.

4.1.4.1 EmailController.php

La classe EmailController.php, serà la API[13] que els desenvolupadors de *frontend* utilitzaran per tal de rebre la informació que hem generat i consta de cinc funcions explicades a continuació i es poden consultar en l'annex 6:

- **list():** Aquesta funció s'encarrega de retornar una llista filtrada per tots els camps d'un correu com: category, from, recipient, subject, body_plain i body_html. I retorna un JSON ordenat per el camp filtrat, amb paginació i tots els camps esmentats anteriorment.
- **addEmail():** Aquesta funció, s'encarrega de crear un nou *email*, afegir el cos del correu en HTML, el fitxer adjunt si s'escau i convertir-ho a JSON per posteriorment, ser enviat per una cua de RabbitMQ.
- **details():** Aquesta funció donat un identificador numèric, retorna del correu buscat, un JSON que contindrà tots els esdeveniments que li han passat a aquest correu.
- **addAttachment():** Aquesta funció, permet afegir a un correu un adjunt.
- **downloadAttachment():** Aquesta funció, permet descarregar donat un identificador de correu numèric, l'adjunt respectiu a l'identificador donat.

4.1.5 Commands

Aquest és un mòdul on es poden definir comandes personalitzades per tal de poder aconseguir amb un ordre, l'execució de diferents serveis o funcions amb facilitat.

4.1.5.1 RabbitMQWait.php

Aquesta classe, situada dins del projecte, crea una comanda(rabbitmq:wait), la qual pot ser cridada. El projecte que l'ha cridat entrarà llavors en mode escolta, esperant un missatge de la cua de RabbitMQ que tindrà assignada. La implementació d'aquesta classe es pot veure en 6.0.4

4.2 Core_Mail_Bus

Aquest component implementat amb PHP, utilitzant Laravel, és l'encarregat de la gestió i comunicació amb els serveis d'Amazon. Aquest, està format per quatre mòduls principals:

- **Routes:** En aquest mòdul és on situen el conjunt d'adreces navegables del component.
- **Models:** És on es definiran les taules de la base de dades així com la configuració d'aquesta.
- **Providers:** En aquest mòdul es defineixen els serveis executables, que ens permeten la comunicació amb les eines externes mencionades en l'apartat 2.
- **Commands:** En aquest apartat, es troba la creació de comandes personalitzades que utilitzarem durant el projecte.

4.2.1 Routes

Routing or router, en desenvolupament web, és un mecanisme on les crides HTTP són dirigides al codi que les tracta. Per simplificar-ho, a l'arxiu que conté rutes li determines el que hauria de passar quan un usuari visita una certa pàgina.

4.2.1.1 channels.php

Aquesta classe es necessita per autoritzar usuaris en canals privats i s'utilitza per a la comunicació de RabbitMQ que tenen els dos projectes.

Aquest fitxer només conté una ruta en el seu interior tal com és pot veure en 6, la ruta esmentada, s'encarrega de rebre una petició HTTP del client i autoritza a l'usuari a accedir a aquest canal. Retornant l'identificador de l'usuari que s'ha autoritzat a accedir.

4.2.1.2 web.php

Tal com es pot veure en l'apartat d'annexos 6, aquesta classe conté una ruta creada per rebre respostes d'Amazon, a més de la inicial que només retorna benvingut quan és consultada.

La ruta anomenada amazonSnsEndpoint, s'encarrega de rebre una petició HTTP de tipus POST del client i usant la funció *parseNotificationPost()* convertirà la resposta d'Amazon a un format JSON el qual és molt més fàcil de tractar.

Una vegada aconseguit, crearà una nova instància de *EmailEvent* que posteriorment enviarà a una cua de RabbitMQ utilitzant la funció *sendRabbitMQResponse()*.

4.2.2 Models

Per tal de comunicar-se eficientment i mantenir coherència en la comunicació entre els dos projectes, els models usats són els mateixos que els del projecte Core_Ilerna_Bus referenciats en aquest apartat anterior 4.1.2

4.2.3 Providers

Els proveïdors de serveis donen les funcionalitats de comunicació necessàries pel projecte. En aquesta secció definirem tots els proveïdors de serveis personalitzats que usarem configurats utilitzant el llenguatge de PHP i les llibreries de les eines esmentades anteriorment.

Així i tot, en aquest apartat els dos projectes també tenen proveïdors de serveis comuns perquè la part de RabbitMQ continua essent comuna en els dos; obviarem l'explicació d'EventServiceProvider.php 4.1.3.2.

Els proveïdors particulars d'aquesta secció són:

- AmazonSesEmailProvider.php
- AmazonSnsProvider.php
- RabbitBusProvider.php

4.2.3.1 AmazonSesEmailProvider.php

Aquesta classe proveïdor com es pot comprovar a l'annex 6.0.1, consta d'una funció anomenada *exec()*. Aquesta, utilitzant el SDK d'Amazon, i a partir d'un *email* i un adjunt, crea una connexió cap a Amazon des d'on, comprovant que tots els camps siguin correctes, afegeix les capçaleres de seguiment al correu i llença missatges d'error pertinents quan falta alguna cosa. Finalment, envia aquest correu a través d'Amazon i retorna el correu enviat.

4.2.3.2 AmazonSnsProvider.php

Aquesta classe proveïdor com es pot comprovar al annex 6.0.2, consta d'una funció anomenada *parseNotificationPost()*, la qual ens permet passar a format JSON un text possibilitant el retorn de la informació en aquest format, i essent molt més fàcil de tractar i utilitzar.

4.2.3.3 RabbitBusProvider.php

Aquesta classe proveïdor com es pot veure en l'annex 6.0.3, conté vàries funcions, de les quals només especificarem *exec()*, ja que és l'única que varia entre els proveïdors de RabbitMQ dels projectes. La funció exec s'encarrega de, donat un missatge codificat amb JSON, executar la funció d' *AmazonSesEmailProvider*, per enviar aquest missatge i retornar un objecte que contindrà totes les dades d'aquest.

4.2.4 Commands

Aquesta, és una carpeta on es poden definir comandes personalitzades per tal de poder aconseguir amb un ordre, l'execució de diferents serveis o funcions amb facilitat.

4.2.4.1 rabbitMQWait.php

Aquesta classe, situada dins del projecte, crea una comanda(rabbitmq:wait), la qual pot ser cridada. El projecte que l'ha cridat entrarà llavors en mode escolta, esperant un missatge de la cua de RabbitMQ que tindrà assignada. La implementació d'aquesta classe es pot veure en 6.0.4 .

5 Avaluació del projecte

Aquest apartat està destinat a analitzar el correcte funcionament de la implementació realitzada, tant a nivell funcional, com de requeriments.
En diferenciarem dues vessants:

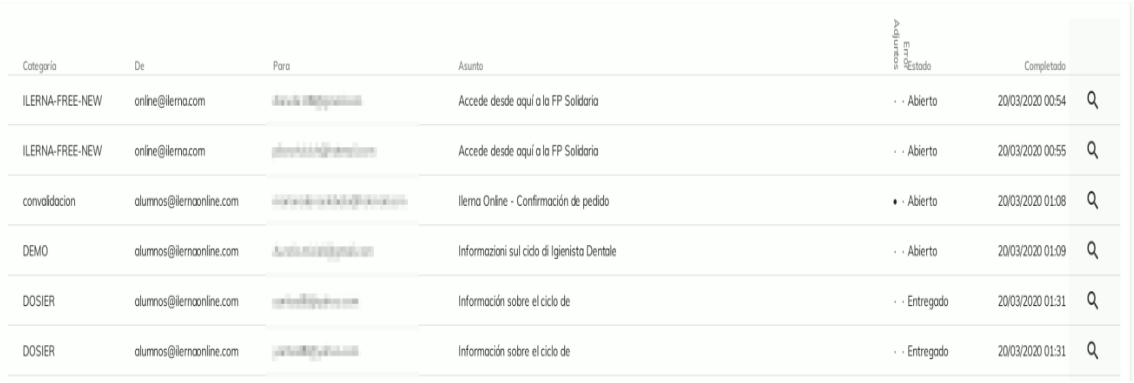
- Assoliment d'objectius
- Asssoliment de requeriments

5.1 Assoliment d'objectius

En aquest apartat es descriurà com s'han assolit els objectius referenciats a l'apartat 1.4. Els quatre primers objectius especificats en l'esmentat apartat 1.4, han estat complerts tal com és s'explica en els capítols 2 i 3 d'aquesta memòria.

El projecte desenvolupat, tal qual es mostrava en la figura 1, té dues sortides: una sortida via web, i una segona sortida mitjançant l'eina Qlick. La figura 5 mostra una visualització de la sortida de la pàgina web. A través d'aquesta figura és pot corroborar com s'han assolit els següents quatre objectius:

1. Fer tests per a la implementació del codi i desplegar l'aplicació en el servidor propi de l'empresa.
2. Integrar l'enviament actual d'*emails* transaccionals amb un sistema de cues.
3. Integrar el sistema de cues amb el núvol, i definir mètriques que capturin la informació que vulguem recollir de la interacció de l'alumne amb l'*email*.
4. Recollir informació de cada correu electrònic transaccional i guardar-la en una base de dades.



Categoría	De	Para	Asunto	Adjuntos	Estado	Completado	
ILERNA-FREE-NEW	online@ilerna.com	[redacted]	Accede desde aquí a la FP Solidaria		Abierto	20/03/2020 00:54	
ILERNA-FREE-NEW	online@ilerna.com	[redacted]	Accede desde aquí a la FP Solidaria		Abierto	20/03/2020 00:55	
convalidacion	alumnos@ilernaonline.com	[redacted]	Ilerna Online - Confirmación de pedido		Abierto	20/03/2020 01:08	
DEMO	alumnos@ilernaonline.com	[redacted]	Informazioni sul ciclo di Igienista Dentale		Abierto	20/03/2020 01:09	
DOSIER	alumnos@ilernaonline.com	[redacted]	Información sobre el ciclo de		Entregado	20/03/2020 01:31	
DOSIER	alumnos@ilernaonline.com	[redacted]	Información sobre el ciclo de		Entregado	20/03/2020 01:31	

Figura 5: Visualització client final

En la figura 5 tenim els correus ordenats per categoria, amb els camps:

- **emissor:** direcció de correu electrònic del que envia el correu.
- **destinatari:** direcció de correu electrònic del receptor.
- **assumpte:** text descriptiu del correu.
- **adjunts:** booleà indicatiu de si hi ha adjunts en el correu o no.

- **error:** booleà indicatiu de si hi ha hagut un error en aquest correu.
- **estat:** descriptiu de l'estat del correu, pot ser: obert, enviat, pendent o desconegut.
- **completat:** data en la qual el correu ha sigut processat per el sistema.

Com es pot veure en la figura 5, l'aplicació ha estat desplegada a producció i es recull, d'acrod als objectius inicials, les dades de cada un dels correus electrònics mostrant-les per la web. La manera de demostrar el total assoliment dels objectius és que l'aplicació ha estat desplegada a producció, els *emails* funcionen amb un sistema de cues, està a més, integrat al núvol i l'aplicació guarda correctament cada correu electrònic enviat en la nostra base de dades MySQL[12].

La figura 6 mostra la sortida que es genera a través de l'eina Qlick. Aquesta ens servirà per corroborar l'últim objectiu, "Gestionar aquesta informació i mostrar-la per obtenir gràfiques i conclusions". Qlick obté la informació directament des de la base de dades. Com es pot veure en la figura 6, la informació retornada per l'eina Qclick integra un conjunt de gràfiques i mètriques que s'expliquen a continuació.

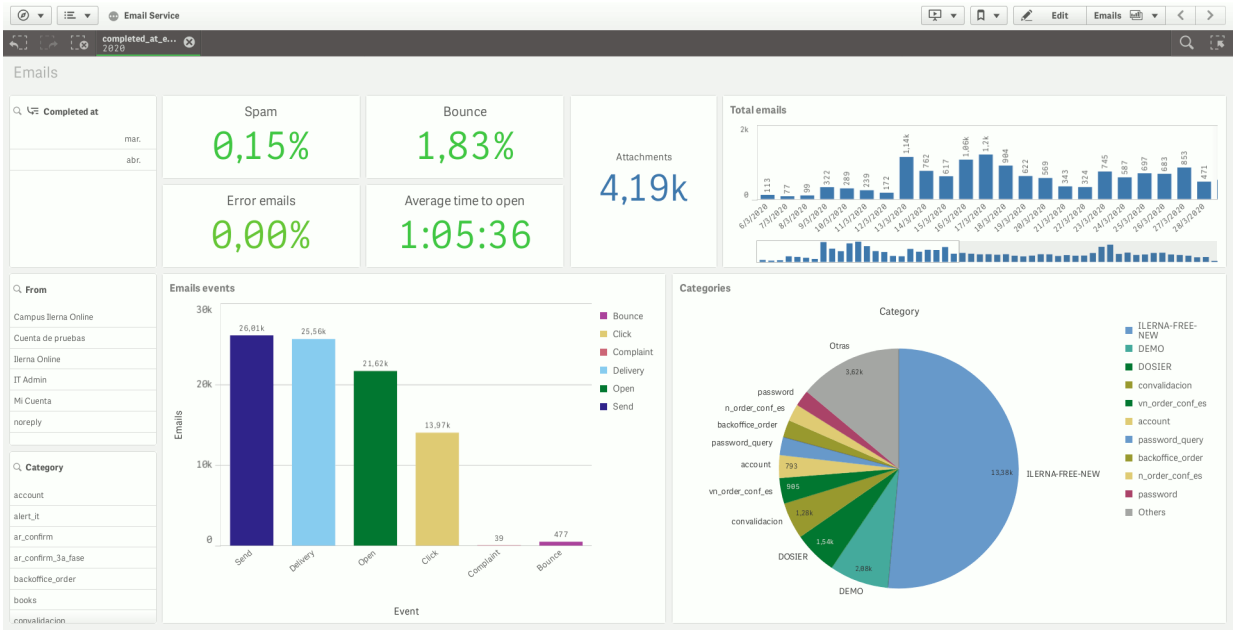


Figura 6: Visualització Qlick amb 3 gràfiques i 4 mètriques.

5.1.1 Gràfiques

- ***Esdeveniments de mail*:** Indica en un gràfic de barres, l'estat de tots els correus electrònics enviats, veient gràficament els oberts, enviats, pendents, així com el nombre de clics en *links*, *spam* i *bounce* definit posteriorment amb més detall.
- **Nombre total d'*emails*:** Indica en un eix de temporalitat, el nombre de correus per dia que s'envia, per tal de poder fer un seguiment acurat i veure la càrrega dels servidors.
- **Categories:** Indica totes les categories d'*email* que s'han enviat fins al moment, mitjançant aquesta gràfica amb un cop d'ull es pot saber quin és el tipus de correu més enviat. En aquest cas es pot veure que la categoria Ilerna-Free-New té un 50% de tot el tràfic, això és degut a que en el moment de completar el projecte, Ilerna es trobava en període d'inscripció.

El millor d'utilitzar aquesta eina de Qlick és que les dades estan vives, això vol dir que si volguéssim filtrar per categoria i per dia, simplement fent clic a la categoria i el dia que vulguem, podríem veure les mateixes dades i gràfiques filtrades per això. I per tant veure l'*spam* d'un dia en una categoria determinada.

5.1.2 Mètriques

- ***Spam*:** Indica el percentatge en correus d'*spam*[13], ja que si és elevat pots tenir restriccions en el servei.
- **Bounce:** Indica el percentatge en correus de *Bounce*[14], ja que si és elevat pots tenir restriccions en el servei.
- **Error *emails*:** Indica el percentatge d'*emails* que contenen errors, o que funcionen anormalment.
- ***Average time to open:*** Indica el temps mitjà d'obertura del correu.

5.2 Assoliment de Requeriments

En aquest apartat descriurem com s'han assolit els requeriments especificats en 3.1.1. Justificarem i analitzarem a partir de les Figures 5 i 7, el correcte assoliment dels requeriments esmentats.

id	creado asunto	email_list	cuerpo	total	Pendiente	Entregado	Abierto	Desconocido	
1	18/03/2020 00:00 Información sobre el ciclo de Técnico Superior en Dietética a distancia	ilerna@example.com	Proving changes in our application.	20	3	12	7	9	
1	18/03/2020 00:00 Información sobre el ciclo de Técnico Superior en Dietética a distancia	ilerna@example.com	Proving changes in our application.	20	3	12	7	9	
1	18/03/2020 00:00 Información sobre el ciclo de Técnico Superior en Dietética a distancia	ilerna@example.com	Proving changes in our application.	20	3	12	7	9	
1	18/03/2020 00:00 Información sobre el ciclo de Técnico Superior en Dietética a distancia	ilerna@example.com	Proving changes in our application.	20	3	12	7	9	
1	18/03/2020 00:00 Información sobre el ciclo de Técnico Superior en Dietética a distancia	ilerna@example.com	Proving changes in our application.	20	3	12	7	9	

Figura 7: Visualització client final

En la figura 7 es pot veure com cada correu que s'envia és emmagatzemat i processat, en aquesta figura es pot veure els arxius adjunts, així com enviaments a múltiples destinataris i el cos de l'email, demostrant l'assoliment dels quatre primers requeriments mostrats a continuació:

1. El sistema haurà de guardar tots els correus que s'enviïn per el mateix.
2. El sistema haurà de tenir una interfície gràfica per poder visualitzar totes les dades.
3. El sistema haurà de permetre adjuntar fitxers i aplicar plantilles per als correus.
4. El sistema haurà de permetre l'enviament a múltiples destinataris, a més, d'enviaments amb remitent ocult i àlies de remitent.

La figura 8 mostra l'eina desenvolupada per filtrar els *emails*. Aquesta eina es correspon amb l'últim requeriment plantejat a 3.1.1 "El sistema haurà de ser capaç de filtrar correus per tipus, correspondència i estat". L'eina, permet filtrar el resultat a partir dels diferents camps, per tal d'obtenir els resultats que nosaltres volem.

Figura 8: Filtre de d'*emails*

Veient com l'aplicació ja en producció es comporta adequadament i podent veure gràficament la informació i filtrar-la pels camps que nosaltres necessitem. Donarem per assolits tots els objectius i requeriments del projecte.

6 Conclusions i futur del projecte

La realització d'aquest treball m'ha servit per guanyar més experiència i poder créixer com a professional en el món laboral d'una empresa. A part d'aprendre sobre com es treballa "avui en dia" en una empresa, també he agafat experiència com a programador aprenent nous llenguatges com PHP i un nou *framework* anomenat Laravel. A més, també he après com funciona tant la part de *backend* com la de *frontend* a l'hora de realitzar aplicacions web.

Podríem dir que m'ha servit per familiaritzar-me més amb l'enginyeria de *software* veient la metodologia que estudiem en acció i com es planifiquen els projectes en un entorn real. La metodologia *Agile* que utilitzen a l'empresa Ilerna Online considero que és molt encertada, perquè per a realitzar projectes de *software* considero que és la millor, tant per la facilitat que dóna tant al programador com al client, ja que facilita la comunicació entre aquestes dues parts. A més l'organització que ens proporciona permet que en tot moment és conegui que està realitzant cadascun dels programadors, quina feina tenen i com solucionen els problemes que tenen, entre altres moltes coses. Crec que per qualsevol departament és molt important el treballar en equip i seguir metodologies com l'*Agile* que potencien aquesta manera d'afrontar els reptes.

En general, he pogut observar d'aprop com treballa un professional i experimentar el treballar en equip en un projecte. Realitzant tant sprints com *sprints plannings, sprint reviews, sprints retrospective* i sobretot *Daily Scrums.*

Referent al llenguatge de programació PHP usat a l'empresa Ilerna Online considero que no és el meu preferit, ja que la sintaxi és més complexa que a altres llenguatges de programació amb les mateixes funcionalitats. Per aquest motiu crec que es podria utilitzar altres llenguatges com Python. En el meu cas, m'hauria agradat Python per l'experiència que tinc programant amb aquest llenguatge. no obstant la realització d'aquests projecte m'ha permès aprendre PHP, el qual és el motor més utilitzat per escriure pàgines web durant els últims anys. A més utilitzant un framework (Laravel) amb l'ORM Eloquent, ha proporcionat una facilitat en la creació web, que no sabria aconseguir amb Python.

Referent a la base de dades, trobo encertada la decisió d'utilitzar MySQL ja que per realitzar aquest projecte es necessita una base de dades mitjana i no necessitem realitzar consultes complexes i llargues. Això sí, al ser una tasca acumulativa, ja que anem guardant moltíssims correus, s'hauria de plantejar canviar en un futur a una millor base de dades com PostgreSQL o alguna que permetés *Big Data*, per tal de poder gestionar la informació sense problemes.

Per concloure, aquest projecte m'ha servit per interioritzar i aprendre d'una manera adequada com es realitza un projecte de *software* en un entorn empresarial, amb una millor noció a l'hora de realitzar un projecte nou com aquest, els passos que s'han de dur a terme i la forma correcta per realitzar-lo. D'aquesta manera, tinc una idea més desenvolupada del treball al qual s'enfronta un veritable professional en l'àmbit informàtic.

Respecte al futur del projecte, crec que és molt polimòrfic i que es podrà continuar desenvolupant per tal de poder gestionar no només els correus transaccionals, sinó tots els correus de l'empresa. Per fer això, només s'haurà d'afegir les funcionalitats necessàries sense haver de canviar la base ja feta, ja que hem complert el principi Obert-Tancat mencionat anteriorment. Això, crea una avantatja de creixement molt important perquè la inversió inicial feta per l'empresa és re-aprofita.

En general penso que el treball realitzat s'ha de considerar satisfactori, ja que és un projecte amb moltes possibilitats i molt viu. Encara hi ha aspectes que es poden millorar una mica més i que per falta d'hores, sobretot pel motiu de la situació d'aquest any, no s'han pogut acabar de perfilar. Però en general, és un projecte del que estic content i espero que ajudi a la comunicació futura entre l'empresa amb els seus alumnes.

Webgrafia

[1] Agile, *Agile Manifesto: https://agilemanifesto.org/iso/ca/manifesto.html*, retrieved at 2020-07-12.

[2] Scrum, *Scrum Guide: https://www.scrumguides.org/docs/scrumguide/v2016/2016-Scrum-Guide-Spanish.pdf#zoom=100*, retrieved at 2020-07-12.

[3] RabbitMQ, *RabbitMQ Guide: https://www.rabbitmq.com/documentation.html*, retrieved at 2020-07-12.

[4] Laravel, *Laravel Documentation: https://laravel.com/docs/7.x/installation*, at 2020-07-12.

[5] Amazon Simple Email Service, *SES Documentation: https://www.rabbitmq.com/documentation.html*, retrieved at 2020-07-12.

[6] AWS SDK PHP, *AWS SDK PHP Documentation: https://docs.aws.amazon.com/es_es/sdkfor-php/?id=docs_gateway*, retrieved at 2020-07-12.

[7] Amazon Simple Notification Service, *Amazon SNS Documentation: https://docs.aws.amazon.com/es_es/sns/?id=docs_gateway*, retrieved at 2020-07-12.

[8] Qlik Community, *Qlik Community Documentation: https://community.qlik.com/t5/QlikEducationDiscussions/DocumentationandReferenceManuals/tdp/1285555*, retrieved at 2020-07-12.

[9] Symfony, *Symfony Web: https://symfony.com/*, retrieved at 2020-07-12.

[10] Open API, *Open API Web: https://swagger.io/specification/*, retrieved at 2020-07-12.

[11] MYSQL, *MYSQL Documentation: https://dev.mysql.com/doc/*, retrieved at 2020-07-12.

Llistat d'acrònims

[1] IT - Information Technology

[2] AMQP - Advanced Message Queuing Protocol

[3] CEO - Chief Executive Officer

[4] IoT - Internet Of Things

[5] HTTP - Hypertext Transfer Protocol

[6] XMPP - Extensible Messaging Presence Protocol

[7] STOMP - Streaming Text Oriented Messaging Protocol

[8] PHP - Hypertext Preprocessor

[9] ORM - Object Relational Mapping

[10] MVC - Model Vista Controlador

[11] AWS - Amazon Web Services

[12] SDK - Software Development Kit

[13] API - Application Programming Interface

Paraules Clau

[1] GET: Operació de HTTP que aconsegueix agafar qualsevol informació que estigui identificada en la URI.

[2] POST: Operació de HTTP que aconsegueix agafar qualsevol informació que estigui identificada en la URI.

[3] Framework: Infraestructura de programari que, en la programació orientada a objectes, facilita la concepció de les aplicacions mitjançant la utilització de biblioteques de classes o generadors de programes.

[4] Correus transaccionals: Per ser qualificats de missatges transaccionals o de relació, l'objectiu principal d'aquestes comunicacions ha de ser "facilitar, completar o confirmar una transacció comercial que el destinatari ha acordat prèviament amb el remitent".

[5] Laravel: És un entorn de treball dissenyat per tal de ser expressiu i elegant, utilitzat sobretot per fer aplicacions web.

[6] Amazon Web Services: Un conjunt de serveis disponibles al núvol propietat d'Amazon.

[7] Qlick: Una aplicació de *bussiness intelligence* que busca entre diversos tipus de dades i les inter-relaciona, permetent a l'usuari mostrar aquestes relacions que a vegades poden ser complexes,de manera clara i entenedora.

[8] RabbitMQ: RabbitMQ és el corredor de missatges de codi obert més utilitzat.

[9] Symfony: Són un conjunt de components de PHP reutilitzables amb varies funcions.

[10] ObertTancat: "Les entitats de *software* (classes, mòduls, funcions, etc.) haurien d'estar obertes a l'extensió, però tancades a la modificació"; això vol dir que el comportament d'aquesta entitat pot ser modificat sense alterar el seu codi font.

[11] Open API: L'especificació OpenAPI (OEA) defineix una interfície estàndard i de llenguatge agnòstic a les API de RESTful que permet tant als humans com als ordinadors descobrir i comprendre les capacitats del servei sense accés a codi font, documentació o mitjançant la inspecció del trànsit de xarxa.

[12] MYSQL: És un sistema de gestió de bases de dades relacional multi-fil i multiusuari, que usa el llenguatge SQL. Suporta de forma nativa PHP.

[13] Spam: Correu marcat com no desitjat o brossa.

[14] Bounce: Reenviament d'un correu fallit, quan per exemple la bústia d'entrada està plena.

[15] Píldora informativa: Una píldora informativa és un petit resum o explicació de cada eina que es fa a tots els membres de l'equip involucrats. Per tal de conèixer millor el seu funcionament.

Annex

Core_Ilerna_Bus

channels.php

```php
<?php

Broadcast::channel('App.User.{id}', function ($user, $id) {
    return (int) $user->id === (int) $id;
});
```

api.php

```php
<?php
Route::middleware('auth:api')-> get('/user', function (Request $request) {
    return $request->user();
});
```

```php
<?php
Route::group(['middleware' => ['jwt:mail:track'], 'prefix' => '/v1/email'], function () {

        Route::get('/', function (Request $request) {
                $handler = new EmailController;
                return $handler->list($request);
        });


        Route::get('/{id}', function (Request $request, $id) {
                $handler = new EmailController;
                return $handler->details($id);
        });

});
```

```php
<?php
Route::group(['middleware' => ['jwt:mail:send'], 'prefix' => '/v1/sendEmail'], function () {

        Route::put('/', function (Request $request) {
                $handler = new EmailController;
                return $handler->addEmail($request);
        });

});
```

```php
<?php
Route::group(['middleware' => ['jwt:mail:attachment'], 'prefix' => '/v1/attachment'], function () {

        Route::get('/{id}', function (Request $request, $id) {
                $handler = new EmailController;
                return $handler->downloadAttachment($id);
        });
```

```
        Route::post('/', function (Request $request) {
                $handler = new EmailController;
                return $handler->addAttachment($request);
        });

});
```

RabbitBusProvider.php

```
<?php
function process_message($message)
{
        echo "Message received: " . $message->body . PHP_EOL;
        RabbitBusProvider::exec($message->body);
}

class RabbitBusProvider extends ServiceProvider
{
    public static $models;

    public function register()
    {
        self::$models = [
            'Email' => \App\Email::class,
            'EmailEvent' => \App\EmailEvent::class,
        ];
    }

public static function createChannel(&$connection)
{
        // dar permisos antes de usar => rabbitmqctl set_permissions user ".*" ".*" ".*"
    $connection = new AMQPStreamConnection(config('rabbitmq.config.host'), intval(config('rabbitmq.c
            , config('rabbitmq.config.username'), config('rabbitmq.config.password'));
    $channel = $connection->channel();
    return $channel;
}

public static function destroyChannel($channel, $connection)
{
    $connection->close();
    $channel->close();
}

 public static function exec($message)
    {
        $object = new \App\IlernaMessage;
        $object->fill(json_decode($message, true));
        $object->received_at = Carbon::now();
        $objectUpdate = \App\IlernaMessage::find($object->id);
        if($objectUpdate !== null)
        {
            $objectUpdate->fill($object->toArray());
            unset($objectUpdate->updated_at);
```

```
            unset($objectUpdate->created_at);
            $objectUpdate->update();
        }
        else
        {
            $objectUpdate = new \App\IlernaMessage;
            $objectUpdate->fill($object->toArray());
            unset($objectUpdate->updated_at);
            unset($objectUpdate->created_at);
            $objectUpdate->save();
        }


        $hModel = self::$models[$object->object_type];
        $subObject = new $hModel;
        if(is_array($object->object_content)) $subObject->fill($object->object_content);
        else $subObject->fill(json_decode($object->object_content, true));
        $subObjectUpdate = $hModel::find($subObject->id);
        if($subObjectUpdate !== null)
        {
            $subObjectUpdate->fill($subObject->toArray());
            unset($subObjectUpdate->updated_at);
            unset($subObjectUpdate->created_at);
            $subObjectUpdate->update();
        }
        else
        {
            $subObjectUpdate = new $hModel;
            $subObjectUpdate->fill($subObject->toArray());
            unset($subObjectUpdate->updated_at);
            unset($subObjectUpdate->created_at);
            $subObjectUpdate->save();
        }
    }

public static function getRabbitMqMessageResponse()
{
    $channel = self::createChannel($connection);

    $channel->queue_declare(config('rabbitmq.config.response_queue'), false, true, false, true);
    $message = $channel->basic_consume(config('rabbitmq.config.response_queue')
            'core_ilerna_bus',false, true, false, false,
            '\App\Providers\process_message');

        while ($channel ->is_consuming()) {
            $channel->wait();
    }
}

 public static function injectRabbitMqMessage($jsondata, $queue_config)
{
    $channel = self::createChannel($connection);
    $channel->queue_declare(config($queue_config), false, true,false, true);
    $msg = new AMQPMessage($jsondata);
    $channel->basic_publish($msg, '', config($queue_config));
```

```
    self::destroyChannel($channel, $connection);
}
```

EventServiceProvider.php

```
<?php
class EventServiceProvider extends ServiceProvider
{
    /**
     * The event listener mappings for the application.
     *
     * @var array
     */
    protected $listen = [
        Registered::class => [
            SendEmailVerificationNotification::class,
        ],
    ];

}
```

EmailController.php

```
<?php
class EmailController extends Controller
{

    function list(Request $request)
    {
        Log::info($request);

        $http_code = 200;
        try {
            $limit = $request->input('limit', 0);

            $emails = Email::when($request->input('category'), function ($query) use ($request) {
                $query->where('category', 'LIKE', '%' . $request->input('category') . '%');
            });

            $emails->when($request->input('from'), function ($query) use ($request) {
                $query->where('from', 'LIKE', '%' . $request->input('from') . '%');
            });

            $emails->when($request->input('recipient'), function ($query) use ($request) {
                $query->where('recipient', 'LIKE', '%' . $request->input('recipient') . '%');
            });

            $emails->when($request->input('subject'), function ($query) use ($request) {
                $query->where('subject', 'LIKE', '%' . $request->input('subject') . '%');
            });

            $emails->when($request->input('body_plain'), function ($query) use ($request) {
```

```
            $query->where('body_plain', 'LIKE', '%' . $request->input('body_plain') . '%');
        });

        $emails->when($request->input('body_html'), function ($query) use ($request) {
            $query->where('body_html', 'LIKE', '%' . $request->input('body_html') . '%');
        });
        if ($request->input('status') !== null) {
            $res = $emails->get()->filter(function ($item) use ($request) {
                return $item->status === $request->input('status');
            });
        } else
        */

        $total_registers = $emails->count();
        if ($limit != 0) $total_pages = $total_registers / $limit;
        else $total_pages = 1;

        $data = $emails->paginate($limit)->makeHidden(['created_at', 'updated_at']);

        $result = array();
        $result['emails'] = $data;
        $result['pagination'] = [
            'total_pages' => ceil($total_pages),
            'total_registers' => $total_registers,
        ];
    } catch (\Exception $e) {
        $error_code = explode('-', (string)Str::uuid())[0];
        $result = ['message' => 'Internal error in server, check logs (error=' . $error_code . '
        Log::error('Internal error [' . $error_code . ']: ' . $e->getMessage() . ' (' . $e->getF
        $http_code = 500;
    }
    return response(json_encode($result, JSON_PRETTY_PRINT), $http_code);

function addEmail(Request $request)
{
    Log::info($request);

    $http_code = 200;
    try {
        $email = new Email;
        $email->fill($request->toArray());
        $email->attachments = $request->attached_ids;
        $email->save();

        $message = new IlernaMessage;
        $message->object_type = 'Email';
        $message->object_content = $email->toJson();
        $message->received_at = \Carbon\Carbon::now();
        $message->save();

        if ($request->attached_ids !== null) {
            $allFiles = array();
            $attachs = explode(',', $email->attachments);
            foreach ($attachs as $att) {
                $id_at = trim($att);
```

```
                $attach_data = Attachment::find($id_at);
                $file = Storage::get('mail-attachments/' . $attach_data->hash . '.' . $attach_data->
                $sst = array();
                $sst['filename'] = $attach_data->filename;
                $sst['filecontent'] = base64_encode($file);
                $allFiles[] = $sst;
            }
            $message->attachs = $allFiles;
        }

        RabbitBusProvider::injectRabbitMqMessage($message->toJson(), 'rabbitmq.config.request_queue_
        $response = ['message' => 'Email accepted in queue',
            'id' => $email->id];
    } catch (\Exception $e) {
        $error_code = explode('-', (string)Str::uuid())[0];
        $response = ['message' => 'addEmail Internal error in server, check logs (error=' . $error_c
        Log::error('Internal error [' . $error_code . ']: ' . $e->getMessage() . ' (' . $e->getFile(
        $http_code = 500;
    }
    return response(json_encode($response, JSON_PRETTY_PRINT), $http_code);
}

function details($id)
{

    $http_code = 200;
    try {
        $email = Email::find($id);
        $emailEvents = EmailEvent::where('ses_id', '=', $email->ses_id)->orderBy('created_at', 'ASC'

        $allAttachments = array();
        if ($email->attachments !== null) {
            $attachs = \explode(',', $email->attachments);
            foreach ($attachs as $attach) {
                $attach = trim($attach);

                $pack = array();
                $pack[] = $attach;
                $pack[] = Attachment::find($attach)->filename;

                $allAttachments[] = $pack;
            }
        }
        $email->attachments = $allAttachments;
        $response = $email->toArray();
        $detailsEvent = array();
        foreach ($emailEvents as $ev) {
            $event_array = $ev->makeHidden(['id', 'updated_at', 'ses_id'])->toArray();
            if ($event_array['link'] === null) unset($event_array['link']);
            $detailsEvent[] = $event_array;
        }
        $response['events'] = $detailsEvent;
    } catch (\Exception $e) {
        $error_code = explode('-', (string)Str::uuid())[0];
```

```
        $response = ['message' => 'Internal error in server, check logs (error=' . $error_code . ')'
        Log::error('Internal error [' . $error_code . ']: ' . $e->getMessage() . ' (' . $e->getFile(
        $http_code = 500;
    }
    return response(json_encode($response, JSON_PRETTY_PRINT), $http_code);
}

function addAttachment(Request $request)
{
    Log::info($request);

    $http_code = 200;
    try {
        $file = $request->all()['file'];

        $attach = new Attachment;
        $attach->hash = (string)Str::uuid();
        $attach->mime_type = $file->extension();
        $attach->filename = $request->input('name');
        $attach->save();

        // almacena el archivo en la nube
        $filename = 'mail-attachments/' . $attach->hash . '.' . $attach->mime_type;
        $uploaded = false;
        do {
            $path = $file->storeAs('mail-attachments', $attach->hash . '.' . $attach->mime_type);
            sleep(1);
            $uploaded = Storage::exists($path);
        } while (!$uploaded);

        $response = ['message' => 'Attached file accepted',
            'id' => $attach->id];
    } catch (\Exception $e) {
        $error_code = explode('-', (string)Str::uuid())[0];
        $response = ['message' => 'addAtt Internal error in server, check logs (error=' . $error_cod
        Log::error('Internal error [' . $error_code . ']: ' . $e->getMessage() . ' (' . $e->getFile(
        $http_code = 500;
    }
    return response(json_encode($response, JSON_PRETTY_PRINT), $http_code);
}
   function downloadAttachment($id)
    {
        $attach = Attachment::find($id);

        return Storage::download('mail-attachments/'.$attach->hash.'.'.$attach->mime_type);
    }
```

Core_Mail_Bus

web.php

```
<?php

Route::get('/', function () {
    return view('welcome');
});

Route::post('/amazonSnsEndpoint', function (Request $request) {

        $bodyContent = $request->getContent();
        if(Str::contains($bodyContent, 'SubscriptionConfirmation'))
        {
                Log::info('Received amazon sns subscription code... ' . $bodyContent);
                return;
        }

        $processedRequest = AmazonSnsProvider::parseNotificationPost($bodyContent);
        $emailEvent = new EmailEvent;
        $emailEvent->ses_id = $processedRequest->mail->messageId;
        $emailEvent->event_type = $processedRequest->eventType;
        if($emailEvent->event_type == 'Click')
                $emailEvent->link = $processedRequest->click->link;

        $message = new IlernaMessage;
        $message->object_type = 'EmailEvent';
        $message->object_content = $emailEvent->toJson();
        $message->completed_at = Carbon::now();
        RabbitBusProvider::sendRabbitMqResponse($message->toJson());
});
```

6.0.1 AmazonSesEmailProvider.php

```
<?php
class AmazonSesEmailProvider extends ServiceProvider
{

    public static function exec(Email $email, $attachs)
    {

        $SesClient = new SesClient([
            'version' => config('amazon_ses.config.version'),
            'region' => config('amazon_ses.config.region'),
            'credentials' => [
                'key' => config('amazon_ses.config.key'),
                'secret' => config('amazon_ses.config.secret'),
            ],
        ]);

        $sender_email = $email->from;

// Replace these sample addresses with the addresses of your recipients. If
```

```
// your account is still in the sandbox, these addresses must be verified.
        $recipient_emails = explode(',', $email->recipient);

// Specify a configuration set. If you do not want to use a configuration
// set, comment the following variable, and the
// 'ConfigurationSetName' => $configuration_set argument below.
        $configuration_set = config('amazon_ses.config.sns_configuration_set');

        $subject = $email->subject;
        $plaintext_body = $email->body_plain;
        $html_body = $email->body_html;
        $char_set = 'UTF-8';
        $toUnlinkAtEnd = array();

        $phpmailer = new PHPMailer;

        if($email->from_name !== null) $phpmailer->setFrom($sender_email, $email->from_name);
        else $phpmailer->setFrom($sender_email);

        // add ccs and bccs
        if($email->cc !== null)
            foreach(\explode(',', $email->cc) as $cc) $phpmailer->AddCC(trim($cc));
        if($email->bcc !== null)
            foreach(\explode(',', $email->bcc) as $bcc)
            {
                if(trim($bcc)!=='welcome@ilernaonline.com') $phpmailer->AddBCC(trim($bcc));
            }

        // multi-recipient
        foreach ($recipient_emails as $rec) $phpmailer->addAddress(trim($rec));

        $isHTML = false;
        if($html_body !== null)
        {
            $isHTML = true;
            $phpmailer->Body = $html_body;
            if($plaintext_body !== null) $phpmailer->AltBody = $plaintext_body;
        }
        else if($html_body === null && $plaintext_body !== null){
            $isHTML = false;
            $phpmailer->Body = $plaintext_body;
        }
        $phpmailer->IsHTML($isHTML);
        $phpmailer->CharSet = $char_set;
        $phpmailer->Subject = $subject;
        if($attachs !== null)
        {
            foreach ($attachs as $att)
            {
                $filename = $att['filename'];
                $filecontent = $att['filecontent'];

                $random_name = '/tmp/' . (string) Str::uuid();
                \file_put_contents($random_name, base64_decode($filecontent));
                $phpmailer->addAttachment($random_name, $filename);
```

```
                $toUnlinkAtEnd[] = $random_name;
            }
        }

        try {
            $phpmailer->addCustomHeader('X-SES-CONFIGURATION-SET', $configuration_set);
            $phpmailer->preSend();
            $message = $phpmailer->getSentMIMEMessage();

            $result = $SesClient->sendRawEmail([
                'Destination' => [
                    'ToAddresses' => $recipient_emails,
                ],
                'ReplyToAddresses' => [$sender_email],
                'Source' => $sender_email,
                'RawMessage' => [
                    'Data' => $message,
                ],
                'ConfigurationSetName' => $configuration_set,
                'Tags' => [
                    [
                        'Name' => 'Category',
                        'Value' => $email->category,
                    ],
                ],
            ]);

            $email->completed_at = Carbon::now();
            $email->ses_id = $result['MessageId'];
        } catch (AwsException $e) {
            // output error message if fails
            echo $e->getMessage();
            echo("The email was not sent. Error message: " . $e->getAwsErrorMessage() . "\n");
            echo "\n";
            $email->error_message = $e->getMessage();
            $email->status = 'Error';
        }

        // clean up attached files and return email
        foreach($toUnlinkAtEnd as $file) unlink($file);
        return $email;
    }
}
```

6.0.2 AmazonSnsProvider.php

```
<?php
class AmazonSnsProvider extends ServiceProvider
{
 public static function parseNotificationPost($postBody)
    {
        $parsedRequest = null;

        foreach(preg_split("/((\r?\n)|(\r\n?))/", $postBody) as $line)
```

```
        {
            $line = trim($line);

            if(Str::startsWith($line, '"Message"'))
            {
                $sep = explode(':', $line);
                unset($sep[0]);
                $sep = implode(':', $sep);
                $sep = str_replace('\"', '"', $sep);
                $sep = str_replace('\n",', '', $sep);

                $sep = explode('"', $sep);
                unset($sep[0]);
                $sep = implode('"', $sep);

                $sep = str_replace('\\"', '"', $sep);

                $parsedRequest = json_decode($sep);
            }
        }

        return $parsedRequest;
    }

}
```

6.0.3 RabbitBusProvider

```
<?php
function process_message($message)
{
        echo "Message received: " . $message->body . PHP_EOL;
    $responseMessage = RabbitBusProvider::exec($message->body);
    RabbitBusProvider::sendRabbitMqResponse($responseMessage->toJson());
}

class RabbitBusProvider extends ServiceProvider
{
    public static $handlersSP;
    public static $handlersModel;

    /**
     * Register services.
     *
     * @return void
     */
    public function register()
    {}

    /**
     * Bootstrap services.
     *
     * @return void
     */
```

```
public function boot()
{}

public static function createChannel(&$connection)
{
    // dar permisos antes de usar => rabbitmqctl set_permissions user ".*" ".*" ".*"
    $connection = new AMQPStreamConnection(config('rabbitmq.config.host'), intval(config('rabbit
        , config('rabbitmq.config.username'), config('rabbitmq.config.password'));
    $channel = $connection->channel();
    return $channel;
}

public static function destroyChannel($channel, $connection)
{
    $connection->close();
    $channel->close();
}


public static function exec($message)
{
    $object = new \App\IlernaMessage;
    $object->fill(json_decode($message, true));
    $object->completed_at = Carbon::now();

    $attachs = null;
    $message_decoded = json_decode($message, true);
    if(array_key_exists('attachs', $message_decoded)
            && $message_decoded['attachs'] !== null)
    {
        $attachs = $message_decoded['attachs'];
    }

    $subObject = new \App\Email;
    $subObject->fill(json_decode($object->object_content, true));

    AmazonSesEmailProvider::exec($subObject, $attachs);

    $object->delegated_id = $subObject->id;
    $object->delegated_error_message = $subObject->error_message;
    $object->completed_at = $subObject->completed_at;
    $object->object_content = $subObject;

    return $object;
}

public static function getRabbitMqMessage()
{
    $channel = self::createChannel($connection);

    $channel->queue_declare(config('rabbitmq.config.request_queue_email'), false, true, false, t
    $message = $channel->basic_consume(config('rabbitmq.config.request_queue_email'), 'core_mail
                                    false, true, false, false, '\App\Providers\process_message')

    while($channel->is_consuming()) {
```

```
            $channel->wait();
        }
    }

    public static function sendRabbitMqResponse($jsondata)
    {
        $channel = self::createChannel($connection);

        $channel->queue_declare(config('rabbitmq.config.response_queue'), false, true, false, true);
        $msg = new AMQPMessage($jsondata);
        $channel->basic_publish($msg, '', config('rabbitmq.config.response_queue'));

        self::destroyChannel($channel, $connection);
    }

}
```

6.0.4 rabbitMQWait.php

```
<?php
class rabbitMqWait extends Command
{
    /**
     * The name and signature of the console command.
     *
     * @var string
     */
    protected $signature = 'rabbitmq:wait';

    /**
     * The console command description.
     *
     * @var string
     */
    protected $description = 'Run daemon for wait and process rabbitmq messages';

    /**
     * Create a new command instance.
     *
     * @return void
     */
    public function __construct()
    {
        parent::__construct();
    }

    /**
     * Execute the console command.
     *
     * @return mixed
     */
    public function handle()
    {
        echo 'Listening the queue...' . PHP_EOL;
```

```
        RabbitBusProvider::getRabbitMqMessage();
    }
}
```